



## P r e l i m i n a r y D o c u m e n t a t i o n

---

# Broadcasting with QuickTime 5



© Apple Computer, Inc. 2001

April 2001

 Apple Computer, Inc.

© 2001 Apple Computer, Inc.  
All rights reserved.

No part of this publication or the software described in it may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Computer, Inc., except in the normal use of the software or to make a backup copy of the software or documentation. The same proprietary and copyright notices must be affixed to any permitted copies as were affixed to the original. This exception does not allow copies to be made for others, whether or not sold, but all of the material purchased (with all backup copies) may be sold, given, or loaned to another person. Under the law, copying includes translating into another language or format. You may use the software on any computer owned by you, but extra copies cannot be made for this purpose.

Printed in the United States of America.

The Apple logo is a trademark of Apple Computer, Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this book. Apple retains all intellectual property rights associated with the technology described in this book. This book is intended to assist application developers to develop applications only for Apple-labeled or Apple-licensed computers.

Every effort has been made to ensure that the information in this manual is accurate. Apple is not responsible for printing or clerical errors.

Apple Computer, Inc.  
1 Infinite Loop  
Cupertino, CA 95014  
408-996-1010

Apple, the Apple logo, LaserWriter, and Macintosh are trademarks of Apple Computer, Inc., registered in the United States and other countries.

Adobe, Acrobat, and PostScript are trademarks of Adobe Systems Incorporated or its subsidiaries and may be registered in certain jurisdictions.

Simultaneously published in the United States and Canada.

**LIMITED WARRANTY ON MEDIA AND REPLACEMENT**

If you discover physical defects in the manual or in the media on which a software product is distributed, ADC will replace the media or manual at no charge to you provided you return the item to be replaced with proof of purchase to ADC.

**ALL IMPLIED WARRANTIES ON THIS MANUAL, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF THE ORIGINAL RETAIL PURCHASE OF THIS PRODUCT.**

Even though Apple has reviewed this manual, **APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

---

## Broadcasting with QuickTime 5 1

### Broadcasting APIs 1

Overview 2

References 3

### Creating a Broadcast From a Live Source 3

Using Data From Different Sources 8

Track Sourcer 9

PushData Sourcer 9

Info Selectors 10

Spatial Parameters 10

Capture settings 11

Compression Settings 11

Get Info Selectors 12

### New Broadcasting APIs 14



# Broadcasting with QuickTime 5



---

QuickTime 5 introduces new broadcasting APIs that allow third-party developers to write broadcasting applications, or to add broadcasting to their existing applications. This document discusses some of the features, capabilities, and usage of these broadcasting APIs.

Note that these APIs have *not* been finalized as of this writing, so the information here is by definition preliminary and subject to change.

If you are interested in participating in a QuickTime seeding program and providing feedback on the new APIs, please send email to [qtswfeedback@apple.com](mailto:qtswfeedback@apple.com).

Your feedback is welcome and encouraged.

---

## Broadcasting APIs

The new QuickTime broadcasting APIs are designed to allow you to create standards-compliant RTP broadcasts. This means that non-QuickTime clients can play back your broadcasts. The new APIs comprise part of the `QTSPresentation` API related to broadcasting and the `QTSSourcer` component API.

(Note that these new broadcasting APIs currently work only on the Mac OS and on Mac OS X.)

Basic broadcasting support of the following is included in the QuickTime 5 release:

- Audio and video streams from the Sequence Grabber.
- Stored movies, i.e., audio and video.
- Text and URLs.

## Broadcasting with QuickTime 5

- Configuration settings with dialogs and info selectors.
- The ability to broadcast for extended periods of time without restarting.
- Reporting of basic statistics, e.g., number of connected users or network stats.
- The ability to broadcast data from different sources (sourcer components).

The QuickTime-supplied sourcers in this release include:

- Sequence Grabber (audio and video only).
- Precompressed media, i.e., sourcer that takes a sample description and sample data.
- Stored movies.

Broadcasting with these new APIs is straightforward enough. The process works as follows:

When you're broadcasting, QuickTime uses the Sequence Grabber to source the media as the default. If you want to generate your own data, your application can create a sample description, with a piece of data, and an accompanying timestamp. Then you call the new broadcast APIs and the data will be broadcast.

## Overview

---

QuickTime streaming is structured so that, at the top level, you have what is defined as a **Presentation**, which is analogous to a QuickTime movie. Within a Presentation, there are a series of streams, which are similar to the tracks in a movie. Streams can be audio, video, or text data.

QuickTime provides a number of API calls for adding and inspecting stream properties, for getting and setting those stream properties, in addition to Presentation properties. This is the same for both sending and receiving streams. When QuickTime *receives* a stream, it creates a Presentation, waits for data, and then creates a stream with that data. When QuickTime *sends* a stream, it creates the stream and has other components within the stream that generate the data and break it into packets.

In its current implementation, you begin by constructing a stream for broadcasting from an SDP (Session Description Protocol) file -- SDP is an industry standard for RTP streaming. (Later implementations of broadcasting may follow a different procedure.) The SDP file specifies a series of streams, i.e., whether they are audio or video, and what compressors will be used for them.

## Broadcasting with QuickTime 5

If you open an SDP file with `QTSNewPresentation`, it creates the Presentation with the number of streams specified in that file.

## References

---

For more information on SDP, refer to <http://ietf.org/rfc/rfc2327.txt>.

For information on packing types, refer to <http://ietf.org/rfc/rfc1890.txt>.

How RTP works is explained in RFC 1889 available at <http://ietf.org/rfc/rfc1889.txt>.

RTSP is explained in RFC 2326 available at <http://ietf.org/rfc/rfc2326.txt>.

## Creating a Broadcast From a Live Source

---

This section describes how you can create a broadcast from a live source, specifying which QuickTime APIs to call in order to start the broadcast. (Note again that the information in this section is preliminary and subject to change. If you are interested in participating in a QuickTime seeding program for broadcasting, please send email to [qtswfeedback@apple.com](mailto:qtswfeedback@apple.com).)

The example begins with the most common case, and then discusses other source types.

To create a broadcast from a live source, you begin by specifying the networking parameters that you want to use. You accomplish this is by using an SDP file or SDP data in memory.

For example:

```
c=IN IP4 224.2.1.2/15/1
m=audio 1000 RTP/AVP 12
m=video 2000 RTP/AVP 101
a=rtpmap:101 H263-1998
```

In this example, all you need to specify are the `c=` and `m=` lines.

The `c=` line defines the destination address, in this case multicast with TTL=15; the `m=` lines define the ports to use (port 1000 for audio, 2000 for video in this

## Broadcasting with QuickTime 5

case). RTP/AVP defines that RTP should be used. The `a=rtpmap` line defines the numbers to be used for dynamic payload types.

If these are not assigned, they will be allocated, starting from 96 when a dynamic packetizer is chosen, and you will need to re-export the sdp file to give to the viewers.

To create a broadcast, fill out `QTSPresParams` like this:

```
struct QTSPresParams {
    UInt32                     version;
    QTSEditListHandle          editList;
    SInt32                     flags;
    TimeScale                  timeScale; /* set to 0 for default
                                              timescale */
    QTSMediaParams *           mediaParams;
    QTSSetNotificationUPP      notificationProc;
    void *                     notificationRefCon;
};

typedef struct QTSPresParams           QTSPresParams;

QTSPresParams             presParams;
QTSMediaParams            mediaParams;
QTSPresentation           presentation = kQTSPresentation;

memset(&presParams, 0, sizeof(presParams));
QTSSInitializeMediaParams(&mediaParams);

mediaParams.v.width = (Fixed)(myWidth<<16);
mediaParams.v.height = (Fixed)(myHeight<<16);
mediaParams.v.gWorld = myGWorld;
mediaParams.v.gdHandle = myGD;

if (sNotificationUPP == NULL)  {
    sNotificationUPP =
        (QTSSetNotificationUPP)NewQTSSetNotificationUPP(_MyNotificationProc);
}

presParams.version = kQTSPresParamsVersion1;
presParams.flags = kQTSAutoModeFlag | kQTSDontShowStatusFlag |
    kQTSSendMediaFlag; presParams.timeScale = kDefaultPresTimeScale;
```

Broadcasting with QuickTime 5

```
presParams.mediaParams = &mediaParams;
presParams.notificationProc = sNotificationUPP;
presParams.notificationRefCon = myRefCon;
```

**To start from an SDP file:**

```
err = QTSNewPresentationFromFile(myFileSpec, &presParams, &presentation);
```

**To start from SDP data in memory:**

```
err = QTSNewPresentationFromData(kQTSSDPDataType, mySDPDataPtr,
                                 mySDPDataLength, &presParams, &presentation);
```

**Once you have a presentation, you should always idle it, using the QTSPresIdle call, since the presentation may be performing network transactions:**

```
QTSPresIdle (QTSPresentation inPresentation,
              QTSPresIdleParams * ioParams);

err = QTSPresIdle(myPresentation,nil); // As is usual with QT, pass 0 or
                                         // nil for a parameter to get the
                                         // default behavior.
```

**If you want to see what the Sequence Grabber is capturing, call QTSPresPreview before starting the broadcast:**

```
QTSPresPreview (QTSPresentation inPresentation,
                QTSSstream inStream,
                const TimeValue64 * inTimeValue,
                Fixed inRate,
                SInt32 inFlags);

err = QTSPresPreview (myPresentation,kQTSA11Streams,nil,kFixed1,0);
```

**To configure the capture source, compression, and packetizer use, call QTSPresSettingsDialog:**

```
QTSPresSettingsDialog (QTSPresentation inPresentation, QTSSstream
                      inStream, SInt32 inFlags, QTSModalFilterUPP
                      inFilterProc, void * inFilterProcRefNum);
```

## Broadcasting with QuickTime 5

The filter proc is for your application to be called back to service idle events and update events.

The Panel filter proc allows you to accept or reject the display of particular settings. Note that settings often interact. If you filter out packetizer settings, but leave compression settings in, the packetizer might change anyway.

If you filter out everything but one panel, it will present that alone.

```
err = QTSPresSettingsDialog (myPresentation, kQTSA11Streams,
                           0,nil,nil,nil,nil);
```

If you supply a `QTSPanelFilterProc`, you'll be called for each panel with a `QTSPanelFilterParams` structure and your refcon. Return true to retain the panel, or false to omit it.

```
struct QTSPanelFilterParams {
    SInt32                           version;
    QTSSstream                        inStream;
    OSType                            inPanelType;
    OSType                            inPanelSubType;
    QTAtomSpec                        details;
};

typedef struct QTSPanelFilterParams QTPanelFilterParams;
/* return true to keep this panel*/
typedef CALLBACK_API( Boolean , QTSPanelFilterProcPtr
)(QTSPanelFilterParams *inParams, void *inRefCon);
```

Once you are ready to begin a broadcast, call `QTSPresPreroll`.

```
QTSPresPreroll (QTSPresentation inPresentation,
                QTSSstream inStream,
                UInt32 inTimeValue,
                Fixed inRate,
                SInt32 inFlags);

err = QTSPresPreroll(myPresentation,kQTSA11Streams,0,kFixed1,0);
```

You should then call `QTSPresIdle` until you get a `PrerollAck` notification via your Notification proc. This allows you to cope with server handshaking, if necessary, for configuration.

Broadcasting with QuickTime 5

```
err = QTSPresIdle(myPresentation,nil);
```

**Once you get the Ack and no error, call QTSPresStart:**

```
QTSPresStart (QTSPresentation inPresentation,
               QTSSetream inStream,
               SInt32 inFlags);
```

```
err = QTSPresStart(myPresentation,kQTSA11Streams,0);
```

**You keep calling idle until the presentation is over; then you call QTSPresStop.**

**When you are finished, dispose the presentation with this call:**

```
QTSDisposePresentation (QTSPresentation inPresentation,
                        SInt32 inFlags);
```

**For someone to be able to see your broadcast, they need to get a copy of your SDP file and open it in QuickTime Player or another RTP-capable application.**

**If you want textual summaries of the broadcast settings to display in your application, you can call QTSPresGetSettingsAsText:**

```
QTSPresGetSettingsAsText (QTSPresentation inPresentation,
                           QTSSetream inStream,
                           SInt32 inFlags,
                           OSType inTextType,
                           Handle * outText,
                           QTSDialogPanelFilterUPP inPanelFilterProc,
                           void * inPanelFilterProcRefNum);

Handle myText=nil;

err = QTSPresGetSettingsAsText (myPresentation, kQTSA11Streams,0,
                                kQTSSettingsTextSummary,&myText, nil, nil);
```

**The panel filter procs can be used to get settings for a subset of the panels, as with the Settings dialog above.**

## Using Data From Different Sources

---

If you want to use data from a source other than the Sequence Grabber, you need to specify a Sourcer. You can do this by using another sdp extension. After each `m=` line, add a line like this:

```
a=x-sourcer:trak
a=x-sourcer:push
a=x-sourcer:none
```

### With the types from this enum

```
enum {
    kQTSSGChannelSourcerType      = FOUR_CHAR_CODE('sgch'),
    kQTSMovieTrackSourcerType    = FOUR_CHAR_CODE('trak'),
    kQTSPushDataSourcerType      = FOUR_CHAR_CODE('push')
};
```

Alternatively, you can specify 'none' above, and add sourcers later by opening a source component instance and using calls like this:

```
QTSPresAddSourcer (QTSPresentation inPresentation,
                    QTSSstream inStream,
                    ComponentInstance inSourcer,
                    SInt32 inFlags);

QTSPresRemoveSourcer (QTSPresentation inPresentation,
                      QTSSstream inStream,
                      ComponentInstance inSourcer,
                      SInt32 inFlags);

QTSPresGetNumSourcers (QTSPresentation inPresentation,
                       QTSSstream inStream);

QTSPresGetIndSourcer (QTSPresentation inPresentation,
                      QTSSstream inStream,
                      UInt32 inIndex,
                      ComponentInstance * outSourcer);
```

## Track Sourcer

---

The Track Sourcer will send data from a track in a QuickTime movie. You need to configure the sourcer by setting the `QTSTrackParams` with a `SetInfo` `kQTSTInfo_Track`, and looping parameters with a.

You can get the current state back in `QTSSourcerTimingParams` with `kQTSTInfo_SourcerTiming` `GetInfo`.

## PushData Sourcer

---

The PushData Sourcer allows you to send out arbitrary data. You provide a sample description and data pointer, and optionally a timestamp. If you don't provide a timestamp, the current time will be used.

```
enum {
    QTSPushDataParamsVersion1 = 1
};

enum {
    QTSPushDataFlag_SampleTimeIsValid = 0x00000001,
    QTSPushDataFlag_DurationIsValid = 0x00000002
};

struct QTSPushDataParams {
    SInt32                                version;
    SInt32                                flags;
    SampleDescriptionHandle                sampleDescription; /* caller owns
                                                                the handle */
    UInt32                                 sampleDescSeed;
    TimeValue64                            sampleTime; /* also set flag if you
                                                                set this */
    TimeValue64                            duration; /* also set flag if you
                                                                set this */
    UInt32                                 dataLength;
    void *                                 dataPtr; /* this does not have to be
                                                                a real macintosh Ptr */
};
typedef struct QTSPushDataParams          QTSPushDataParams;
```

## Info Selectors

---

```
/* info selectors for sourcers - get and set */
kQTSInfo_Track = FOUR_CHAR_CODE('trak'), /* QTSTrackParams* */
kQTSInfo_Loop = FOUR_CHAR_CODE('loop'), /* QTSLoopParams* */
kQTSInfo_SourcerTiming = FOUR_CHAR_CODE('stim'), /*
                                                QTSSourcerTimingParams* */
the above 3 are for the Track sourcer
```

```
kQTSInfo_TargetFrameRate = FOUR_CHAR_CODE('tfps'), /*
                                                Fixed * in frames per second */
```

This defines the target frame rate from the Sg Sourcer.

```
kQTSInfo_PushData = FOUR_CHAR_CODE('push'), /* QTSPushDataParams* */
```

This lets you send data into a pushdata.

```
kQTSInfo_SourcerCallbackProc = FOUR_CHAR_CODE('scbp'), /*
                                                QTSSourcerCallbackProcParams* */
```

This allows you to get called back by a track sourcer when it is completed.

## Spatial Parameters

---

Overall, the Presentation has Media Parameters, which define how it is locally displayed. In the absence of any other information, what is passed in for `QTSNewPresentation` will initialize Sourcer values as well.

The values here can be retrieved with `QTSPresGetxxx`. For example:

```
QTSPresGetMatrix(pres, kQTSAllStreams,&outMatrix);
QTSPresGetDimensions(pres, kQTSAllStreams,&outWidth,&outHeight);
```

If you don't pass `kQTSAllStreams`, this will go down to the Sourcers for that stream instead.

In addition, for Sequence Grabber sources, there are further selectors for cropping, which currently call through to the Sequence Grabber:

- `kQTSInfo_FullInputRect` is a get-only call to find out the maximum rectangle of the source.

## Broadcasting with QuickTime 5

- `kQTSInfo_CroppedInputRect` is a get/set call for the cropped rectangle relative to the rectangle above.

**Capture settings**

---

```
kQTSInfo_InputDeviceName = FOUR_CHAR_CODE('innm'), /* Handle* */
kQTSInfo_InputSourceName = FOUR_CHAR_CODE('srnm'), /* Handle* */
```

**The following calls map through to the VDIG settings:**

```
kQTSInfo_VideoHue = FOUR_CHAR_CODE('hue '), /* UInt16* */
kQTSInfo_VideoSaturation = FOUR_CHAR_CODE('satr'), /* UInt16* */
kQTSInfo_VideoContrast = FOUR_CHAR_CODE('trst'), /* UInt16* */
kQTSInfo_VideoBrightness = FOUR_CHAR_CODE('brit'), /* UInt16* */
kQTSInfo_VideoSharpness = FOUR_CHAR_CODE('shrp') /* UInt16* */
```

**The following calls map through to Sound Manager calls:**

```
kQTSInfo_AudioAutoGainOnOff = FOUR_CHAR_CODE('agc '), /*
                           Boolean* - error if unavailable*/
kQTSInfo_AudioGain = FOUR_CHAR_CODE('gain'), /*
                           Fixed* kFixed1 is unity gain */
```

**Compression Settings**

---

```
kQTSInfo_TargetFrameRate = FOUR_CHAR_CODE('tfps'), /*
                           Fixed * in frames per second */
kQTSInfo_TargetDataRate = FOUR_CHAR_CODE('tdrt'), /*
                           UInt32 * in bytes per second */
```

**The underlying stdCompression selectors have been exposed, so you can find and change compression parameters:**

**Visual:**

```
kQTSInfo_SpatialSettings = FOUR_CHAR_CODE('sptl'), /*
                           pointer to SCSSpatialSettings struct*/
kQTSInfo_TemporalSettings = FOUR_CHAR_CODE('tprl'), /*
                           pointer to SCTemporalSettings struct*/
kQTSInfo_DataRateSettings = FOUR_CHAR_CODE('drat'), /*
```

## Broadcasting with QuickTime 5

```

pointer to SCDataRateSettings struct*/
kQTSSource_CodecFlags = FOUR_CHAR_CODE('cflg'), /*
pointer to CodecFlags*/
kQTSSource_CodecSettings = FOUR_CHAR_CODE('cdec'), /*
pointer to Handle*/
kQTSSource_ForceKeyValue = FOUR_CHAR_CODE('ksim'), /*
pointer to long*/

```

**Audio:**

```

kQTSSource_SoundSampleRate = FOUR_CHAR_CODE('ssrt'), /*
pointer to UnsignedFixed*/
kQTSSource_SoundSampleSize = FOUR_CHAR_CODE('ssss'), /*
pointer to short*/
kQTSSource_SoundChannelCount = FOUR_CHAR_CODE('sscc'), /*
pointer to short*/
kQTSSource_SoundCompression = FOUR_CHAR_CODE('ssct'), /*
pointer to OSType*/

```

**Codec list for both (setting this can be used to constrain the choice of Codecs):**

```

kQTSSource_CompressionList = FOUR_CHAR_CODE('ctyl'), /*
pointer to OSType Handle*/
kQTSSource_SGChannel = FOUR_CHAR_CODE('sgch'), /*
SGChannel* */

```

This will directly give the SGChannel; its use is discouraged in favor of the selectors above.

**Get Info Selectors**


---

/\* get and set \*/

```

enum {
    kQTSSoundLevelMeteringEnabledInfo = FOUR_CHAR_CODE('mtrn'), /*
Boolean* */
    kQTSSoundLevelMeterInfo = FOUR_CHAR_CODE('levm'), /*
LevelMeterInfoPtr */
    kQTSSourceTrackIDInfo = FOUR_CHAR_CODE('otid'), /* UInt32* */
    kQTSSourceLayerInfo = FOUR_CHAR_CODE('olyr'), /* UInt16* */
    kQTSSourceLanguageInfo = FOUR_CHAR_CODE('olng'), /* UInt16* */
}

```

## Broadcasting with QuickTime 5

```

kQTSSourceTrackFlagsInfo = FOUR_CHAR_CODE('otfl'), /* SInt32 */
kQTSSourceDimensionsInfo = FOUR_CHAR_CODE('odim'), /*
QTSDimensionParams */
kQTSSourceVolumesInfo = FOUR_CHAR_CODE('ovol'), /*
QTSVolumesParams */
kQTSSourceMatrixInfo = FOUR_CHAR_CODE('omat'), /*
MatrixRecord */
kQTSSourceClipRectInfo = FOUR_CHAR_CODE('oclp'), /* Rect */
kQTSSourceGraphicsModeInfo = FOUR_CHAR_CODE('ogrm'), /*
QTSGraphicsModeParams */
kQTSSourceScaleInfo = FOUR_CHAR_CODE('oscl'), /* Point */
kQTSSourceBoundingRectInfo = FOUR_CHAR_CODE('orct'), /* Rect */
kQTSSourceUserDataInfo = FOUR_CHAR_CODE('oudt'), /* UserData */
kQTSSourceInputMapInfo = FOUR_CHAR_CODE('oimp'), /*
QTAtomContainer */
};

/* get only */
enum {
QTSStatisticsParams */
kQTSMInStatusDimensionsInfo = FOUR_CHAR_CODE('mstd'), /*
QTSDimensionParams */
kQTSNormalStatusDimensionsInfo = FOUR_CHAR_CODE('nstd'), /*
QTSDimensionParams */
kQTSTotalDataRateInfo = FOUR_CHAR_CODE('drtt'), /*
UInt32*, add to what's there */
kQTSTotalDataRateInInfo = FOUR_CHAR_CODE('drti'), /*
UInt32*, add to what's there */
kQTSTotalDataRateOutInfo = FOUR_CHAR_CODE('drto'), /*
UInt32*, add to what's there */
kTSLostPercentInfo = FOUR_CHAR_CODE('lpct'), /*
QTSLostPercentParams*, add to what's there */
kQTSSNumViewersInfo = FOUR_CHAR_CODE('nviw'), /* UInt32 */
kQTSMediaTypeInfo = FOUR_CHAR_CODE('mtyp'), /* OSType */
kQTSSNameInfo = FOUR_CHAR_CODE('name'), /* QTSSNameParams */

```

## New Broadcasting APIs

---

**The following is a preliminary list -- from `QuickTimeStreaming.h` and `QTStreamingComponents.h` -- of some of the broadcasting APIs that are new in QuickTime 5.0.1:**

```
QTSInitializeMediaParams
QTSNewPresentationFromData
QTSNewPresentationFromDataRef
QTSNewPresentationFromFile
QTSNewSourcer
QTSPresAddSourcer
QTSPresExport
QTSPresGetIndSourcer
QTSPresGetNumSourcers
QTSPresGetSettingsAsText
QTSPresPreview
QTSPresRemoveSourcer
QTSPresSettingsDialog
QTSPresSettingsDialogWithFilters
QTSSourcerGetEnable
QTSSourcerGetInfo
QTSSourcerGetTimeScale
QTSSourcerIdle
QTSSourcerInitialize
QTSSourcerSetEnable
QTSSourcerSetInfo
QTSSourcerSetTimeScale
```

The next revision of this document will discuss in detail how these new calls work and how you can take advantage of them in writing broadcasting applications or adding broadcasting to your existing applications.

I N D E X